REMARKS

Claims 1-53 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 103(a) Rejection:

The Examiner rejected claims 1, 2, 6-9, 22, 23, 27, 30, 31 and 43-46 under 35 U.S.C. § 103(a) as being unpatentable over Sundaresan (U.S. Patent. 6,569,207) in view of Cavanaugh, et al. (U.S. Patent No. 5,991,823) (hereinafter "Cavanaugh"). Applicants respectfully traverse this rejection in light of the following remarks.

In regard to claim 1, contrary to the Examiner's assertion, Sundaresan in view of Cavanaugh does not teach or suggest generating a computer programming language object from a data representation language representation of the object, wherein the object is an instance of a class in the computer programming language. The Examiner refers to the teachings in Sundaresan regarding instantiating objects from class specifications generated based on XML schemas. The Examiner cites passages of Sundaresan (column 9, lines 7-16, and lines 23-38) describing the creation of Java class specifications from XML schemas and instantiating objects based on those class specifications and on XML documents conforming to the XML schemas. However, the XML documents in Sundaresan are not representations of objects that are instances of classes in a computer programming language. Sundaresan teaches that a Java Bean object may be generated (based on Java Bean specification generated from an XML schema) to allow reading and manipulating data in an XML document. Like other conventional uses of XML, Sundaresan teaches *XML to represent data*, not programming language objects. Sundaresan does not describe XML document 114 as a data representation language representation of a computer programming language object. Instead, the XML document in Sundaresan is just a conventional XML document. Sundaresan does instantiate an object from the XML document. But prior to this instantiation the original XML document did not contain a *representation* of a computer

programming language object. Sundaresan only describes the XML document as a traditional XML document representing data, not a computer programming language object.

Sundaresan states, "the attributes and elements of the DTD become the properties of the corresponding Java Bean (Sundaresan, column 5, lines 1-3). Specifically, Sundaresan teaches that his BeanMaker 112 generates a Java class with a name built from an XML DTD and also teaches allowing a user to specify the name of a class (Sundaresan, column 5, lies 24-34). Sundaresan teaches the same sort of name determination and generation for each element in an XML DTD. The information to generate an object in Sundaresan clearly does not come from the XML document itself. Thus, Sundaresan's XML documents cannot be representations of programming language objects. If they were, there would be no need to generate class specifications from XML schemas, since a representation of a programming language object would already include such information.

The teachings of Sundaresan are very similar to the teachings of Jain et al. (U.S. Publication No. 2002/0073091) which the Examiner relied upon in the previous Office Action. Both Jain and Sundaresan determine class specifications from an XML DTD and then generate objects according to those class specifications and data from an XML document.

In further regard to claim 1, contrary to the Examiner's assertion, Sundaresan does not teach removing the computer programming language object. The remove method cited by the Examiner (Sundaresan, column 8, lines 55-65) refers to a remove method that is generated, along with get, set, has, and add methods, for "each attribute or element" of an XML document and according to an XML schema. Specifically, Sundaresan teaches that remove methods are generated for properties whose occurrence is optional and that remove methods remove a property instance at a particular index (Sundaresan, column 8, lines 45-53). The remove methods taught by Sundaresan have nothing to do with removing the computer programming language object, but instead

manipulate individual data elements of an XML document through a generated Java Beans object.

The Examiner relies upon Cavanaugh to teach deleting the computer programming language object in response to the user terminating the accessing of a client device. The Examiner cites passages of Cavanaugh (column 15, lines 9-23, and column 13, lines 28-51) that describe Cavanaugh's use of both a deactivate implementation function and a shutdown function. However, Cavanaugh does not teach removing an object in response to a user terminating the accessing of a client computer. Instead, Cavanaugh teaches that the deactivate implementation function is called when the system determines that no more invocations of a particular implementation are desired. The second portion of Cavanaugh cited by the Examiner refers to a shutdown function "used to shutdown a particular implementation definition by removing all servant objects" (Cavanaugh, column 13, lines 46-48). Specifically, Cavanaugh teaches that an object request broker (ORB) may wish to shut down a server using the shutdown function (Cavanaugh, column 15, lines 26-30) and that the shutdown function is used "to shutdown all implementation of objects within the ORB" (Cavanaugh, column 16, lines 43-44). Nowhere does Cavanaugh describe either his deactivate implementation function or his shutdown function as deleting a computer programming language object in response to a user terminating the accessing of a client device. Thus, Sundaresan and Cavanaugh, singly and in combination, fail to teach deleting the computer programming language object in response to a user terminating the accessing of a client device.

In light of the above remarks, Applicants assert that the rejection of claim 1 is not supported by the teachings of the cited art. As such, Applicants respectfully request removal of the 35 U.S.C. § 103(a) rejection of claim 1. Similar remarks apply in regard to claims 22 and 43.

Regarding claim 2, contrary to the Examiner's assertion, Sundaresan in view of Cavanaugh fails to teach the client device receiving a message in the data representation language from a service device in the distributed computing environment prior to said

generating a computer programming language object, wherein the *message includes* the data representation language representation of the object. The Examiner cites column 12, lines 20-49 of Sundaresan; however, this passage of Sundaresan does not mention a message including a data representation language representation of a computer programming language object. Instead, the cited portion of Sundaresan describes the embedding of Java Bean specifications in script files. Specifically, Sundaresan teaches various ways of using XML specifications embedded in script files, HTML pages, servlet and Java server pages. However, as noted above regarding claim 1, the XML specifications described by Sundaresan are not data representation language representations of computer programming language objects. Thus, the combination of Sundaresan in view of Cavanaugh fails to mention anything regarding receiving a message in a data representation language wherein the message includes a data representation language representation of an object.

In light of the above remarks, Applicants assert that the rejection of claim 2 is not supported by the teachings of the cited art and removal thereof is respectfully requested. Similar remarks apply in regard to claims 23 and 44.

Claims 3-5, 10-21, 24-26, 28, 29, 32-42 and 47-53 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Sundaresan and Cavanaugh in view of Wu (U.S. Patent 5,774,551). Applicants respectfully traverse this rejection in light of the following remarks.

Regarding claim 10, contrary to the Examiner's contention, Sundaresan in view of Cavanaugh in further view of Wu fails to teach the client device receiving a message in a data representation language from a service device in the distributed computing environment, wherein the message includes a data representation language representation of a computer programming language object. The Examiner cites column 12, lines 20-49 of Sundaresan, however, this passage of Sundaresan does not mention a message including a data representation language representation of a computer programming language object. Instead, the cited portion of Sundaresan describes the embedding of

Java Bean specifications in script files. Specifically, Sundaresan teaches various ways of using XML specifications embedded in script files, HTML pages, servlet and Java server pages. However, the XML specifications described by Sundaresan are not data representation language representations of computer programming language objects. Please see the arguments presented above regarding Sundaresan's generation of Java Bean objects in the Examiner's rejection of claim 1, above.

Thus, Sundaresan teaches that an XML document and DTD may be embedded in HTML and that a Java Bean for that XML document may be automatically generated to allow the Java Beans to "interact with the surrounding scripts and with other components" (Sundaresan, column 12, lines 46-49), but fails to teach receiving a message in a data representation language including a data representation language representation of a computer programming language object.

Furthermore, contrary to the Examiner's assertion, Sundaresan in view of Cavanaugh in further view of Wu fails to teach determining if the user has access rights to the computer programming language object. The Examiner relies on Wu and refers to Wu's teachings regarding granting access to a computer system and creating user credentials (column 19, lines 15-45). However, Wu does not teach that such authentication services can be used to determine whether a user has access rights to a computer programming language object. Instead, Wu teaches multiple authentication services that provide different types of authentication schemes for establishing and verifying an identity of a user attempting to access a computer system in general (Wu, column 15, lines 54-63). Nowhere does Wu mention determining access rights to individual computer programming language objects. Authenticating a user before granting access to a computer does not include or imply determining whether the user has access rights to a computer programming language object.

Additionally, as shown above regarding claim 1, Sundaresan in view of Cavanaugh fails to teach generating an object from the data representation language representation of the object, wherein the object is an instance of a class in the computer

programming language. The remarks above regarding claim 1 apply here as well. Further, Wu fails to overcome any of the deficiencies of Sundaresan and Cavanaugh regarding generating the object from the data representation language representation of the object.

Contrary to the Examiner's contentions, the combination of Sundaresan, Cavanaugh, and Wu would only result in a system that uses the various authentication services of Wu to determine whether a user has access to a computer system on which Sundaresan's method of automatically generating Java Bean objects from XML schemas is performed and that uses the sub-contract method of Cavanaugh to control selected groupings of features or services.

In light of the above remarks, Applicants assert that the rejection of claim 10 is not supported by the teachings of the cited art. As such, Applicants respectfully request removal of the 35 U.S.C. § 103(a) rejection of claim 10. Similar remarks apply in regard to claims 32 and 47.

Regarding claim 11, contrary to the Examiner's assertion, Sundaresan in view of Cavanaugh in further view of Wu fails to teach wherein the message further includes access information for the computer programming language object, wherein said determining if the user has access rights to the computer programming language object uses the access information. The Examiner claims that Sundaresan teaches a message including access information for the computer programming language object and cites column 9, lines 5-16. Specifically, the Examiner points to Sundaresan's Java class specifications 116. However, Sundaresan does not disclose anything regarding access information, nor about any Java class specifications including access information. Additionally, the Examiner's cited passage of Sundaresan refers to how his Bean Maker 112 creates Java class specifications 116 from XML schemas 114, but does not mention either any message, any access information, or any message including access information.

The Examiner relies up on Wu to teach wherein the determining if the user has access rights to the computer programming language object uses the access information and cites column 19, lines 1-15 of Wu. However, the cited passage of Wu fails to mention using accessing information included in a data representation language message when determining if a user has access rights to a computer programming language object. Instead, Wu teaches a unified login for multiple authentication services for gaining access to a computer system. Wu does not teach receiving a message in a data representation language that includes access information for a computer programming language object, nor about use such access information when determining if a user has access to the computer programming language object. In fact, Wu fails to mention anything about determining whether a user has access to a computer programming language object. Wu is concerned with user access to a computer system, not individual computer programming language objects. Furthermore, the Examiner's cited passage describes the use of a system entry service 107, which does not send messages in a data representation language.

Additionally, Cavanaugh fails to overcome the above noted deficiencies of Sundaresan and Wu. Thus, Sundaresan in view of Cavanaugh in further view of Wu fails to teach wherein the message further includes access information for the computer programming language object, wherein said determining if the user has access rights to the computer programming language object uses the access information. The rejection of claim 11 is therefore unsupported by the cited art and removal thereof is respectfully requested. Similar remarks apply in regard to claims 33 and 48.

Regarding claim 12, contrary to the Examiner's assertion, Sundaresan in view of Cavanaugh in further view of Wu does not teach removing the computer programming language object in response to the user terminating the accessing of the client device. The Examiner relies upon Cavanaugh and cites passages column 15, lines 9-23, and column 13, lines 28-51 that describe Cavanaugh's use of both a deactivate implementation function and a shutdown function. However, as described above regarding claim 1, Cavanaugh does not teach removing an object in response to a user

terminating the accessing of a client computer. Instead, Cavanaugh teaches that the deactivate implementation function is called when the system determines that no more invocations of a particular implementation are desired. The second portion of Cavanaugh cited by the Examiner refers to shutdown function "used to shutdown a particular implementation definition by removing all servant objects if necessary" (Cavanaugh, column 13, lines 46-48). Specifically, Cavanaugh teaches that an object request broker (ORB) may wish to shut down a server using the shutdown function (Cavanaugh, column 15, lines 26-30) and that the shutdown function is used "to shutdown all implementation of objects within the ORB" (Cavanaugh, column 16, lines 43-44). Nowhere does Cavanaugh describe either his deactivate implementation function or his shutdown function as deleting a computer programming language object in response to a user terminating the accessing of a client device.

Thus, the rejection of claim 12 is not supported by the cited art and therefore its removal is respectfully requested. Similar remarks apply in regard to claims 34 and 49.

Regarding claim 16, despite the Examiner's contention, Sundaresan as modified fails to teach storing the computer programming language object in response to said terminating access. The Examiner cites column 4, lines 37-45 of Sundaresan, but this portion of Sundaresan only describes how the instructions that make up his BeanMaker 112 may generally be stored and/or communicated on computer media, but does not have anything to do with storing a computer programming language object in response to a user terminating the accessing of a client device. In fact, the cited passage of Sundaresan only mentions computer instructions that cause a computer to perform the steps for his invention. This general statement by Sundaresan is clearly directed to the distribution of his BeanMaker 112 on various types of computer media and has nothing to do with storing individual computer programming language objects.

Additionally, the Examiner cites the unified logout process at column 19, line 57 – column 20, line 8 of Wu. However, this portion of Wu describes only destroying various authentication credential when a user logs out of a system. Nowhere does Wu

mention storing a computer programming language object as part of his unified logout process.

Thus, the rejection of claim 16 is not supported by the cited art and therefore its removal is respectfully requested. Similar remarks apply in regard to claims 37 and 51.

Applicants also assert that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any extension of time (under 37 C.F.R. § 1.136) is necessary to prevent the above referenced application from becoming abandoned, Applicants hereby petition for such extension. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-47300/RCK.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

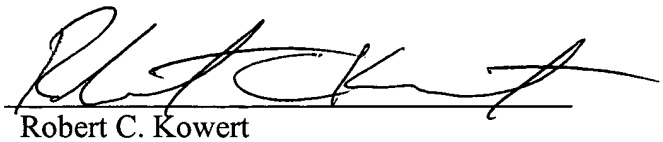
☐ Petition for Extension of Time

☐ Notice of Change of Address

☐ Fee Authorization Form authorizing a deposit account debit in the amount of $ for fees ( ).

☐ Other:

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: November 11, 2004